Data Models & Query Languages: Milestone 2

Project Name: Campus Recruitment Database ::: Team Name: Seekers

Katipelly Shreya Reddy
UBID: skatipel
Member1

Desireddy Sai Sankeerthana
UBID: saisanke
Member2

Abhigna Sree Tumati
UBID: abhignas
Member3

I. Problem Statement

With each passing year, thousands of students will be graduating, and are exposed to various job offers in various companies based on their educational qualifications. To keep track of each person's data and infer information on how the records of the students in the campus are varying and to take any measures to improve the statistics and to understand how each factor is affecting the placements opportunities each year, The college officials must be able to understand how things are working for students in campus. And maintaining such huge data in a file system will not be of much use since we it is hard to infer information from a file system. So, in here, database would be of good use and helps them to get a good picture of everything related to campus placements and how their educational background is helping them to grab better opportunities. We will be using a database instead of an excel sheet to store the data of our campus recruitment information because in a database we can define relations between various schemas and retrieve data easily for understanding. We can restrict the type of data entered in the records by 'Data Integrity' so that we can avoid mistakes while handling large data. In a database we can also use the concept of 'Referential Integrity' which ensures there is a value in another table which can be referenced. A database can handle large amount of data which a excel sheet cannot. A database can be accessed by multiple users at a given time whereas a excel can only be used by a single person at any point of time. If we want to secure our data and provide privileges to users to perform various operations on our data, then databases are very efficient where it provides security to our data by applying certain restrictions on who can use our data which an excel sheet fails to provide.

II. Target User

The target audience and the administrator for the database would be the College officials. Every College wants to be recognized as one of the prestigious one's in terms of the education it offers to the students. To be recognized as one, the college must be providing the students with a great academic program. And the campus placement percentage would be considered as metric since a good program should be able to equip the students with skills to get ready for the industry. So having a database to store all such data and inferring information could help them in getting an idea of what is going well and what needs improvisation. This way, they can make necessary changes to their program to keep the student's industry ready and helps the college officials to achieve their end goal to provide their best for the students.

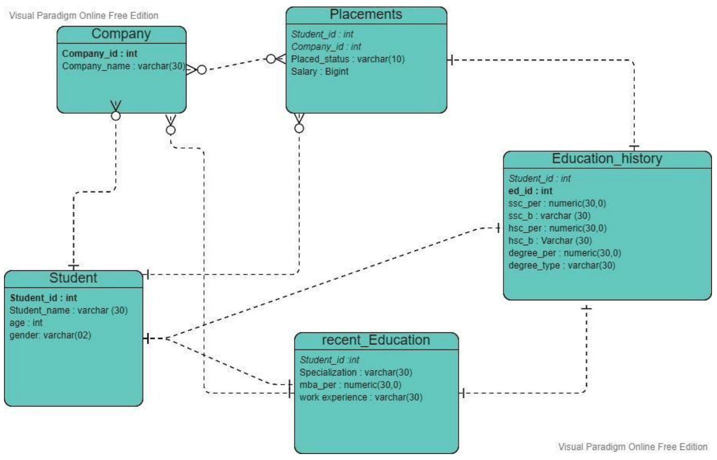


Fig. 1. E-R Diagram

III. Details on Schema

Final Schema of the implemented database is the same as the initial one.

A. Action on the existing foreign keys in case of deletion of a primary key value

Student Id : Delete cascade Upon the deletion of a student details in the student table, no further data regarding the student is necessary anymore. So the action taken here is delete cascade Company Id : No action The deletion of a company tuple from the company table in the present doesn't have to effect the rest of the student data since we are having records of students past data. So in this case we are taking no action

IV. Dependencies

All our tables are already in BCNF and following are the FD's for each table.

A. Company

Company (company_ id, company_name)
1) company_id $\longrightarrow$ company_name

B. Placements

Placements (student_id, company_id, placed_status, salary)
1) student_id, company_id $\longrightarrow$ placed_status
2) student_id, company_id $\longrightarrow$ salary

C. Student

Student (student_id, student_name, age, gender)
1) student_id $\longrightarrow$ student_name
2) student_id $\longrightarrow$ age
3) student_id $\longrightarrow$ gender

D. Recent_Education

Recent_Education (student_id, specialization, mba_per, work_experience)
1) student_id $\longrightarrow$ specialization
2) student_id $\longrightarrow$ mba_per
3) student_id $\longrightarrow$ work_experience

E. Education_History

Education_History (student_id, ed_id, ssc_per, ssc_b, hsc_per, hsc_b, degree_per, degree_type)
1) ed_id $\longrightarrow$ student_id
2) ed_id $\longrightarrow$ ed_id
3) ed_id $\longrightarrow$ ssc_per
4) ed_id $\longrightarrow$ ssc_b
5) ed_id $\longrightarrow$ hsc_per
6) ed_id $\longrightarrow$ hsc_b
7) ed_id $\longrightarrow$ degree_per
8) ed_id $\longrightarrow$ degree_type

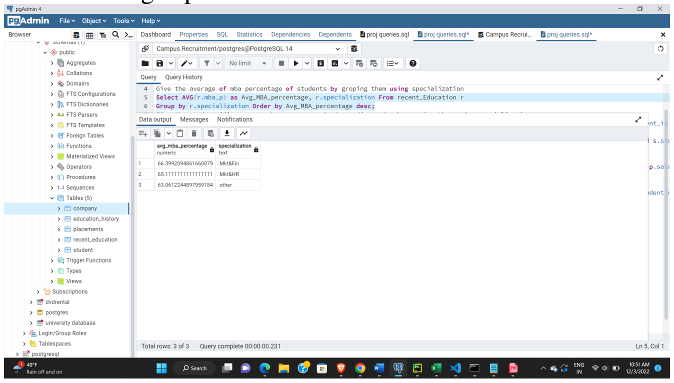
V. Query Execution

We've executed multiple queries on the implemented Campus Recruitment Database including various concepts like subqueries, joins, order by, group by, aggregate functions etc.
1. Selection
2. Deletion
3. Insertion
4. Updating

The following are the screenshots of few of them:

A. Selection Queries

1)Query to find id's of students who got MBA percentage less than 80 and got placed.

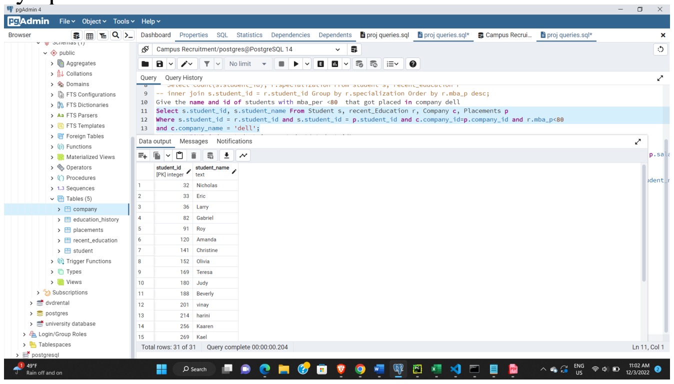


```
Give the average of mba percentage of students by groping them using specialization
Select AVG(r.mba_p) as Avg_MBA_percentage, r.specialization From recent_Education r
Group by r.specialization Order by Avg_MBA_percentage desc;
```

```
   avg_mba_percentage   specialization
   numeric              text
1  66.3992094861660079  Mkt&Fin
2  65.1111111111111111  Mkt&HR
3  63.0612244897959184  other

Total rows: 3 of 3    Query complete 00:00:00.231
```

Fig. 2.

2) Query to find Average MBA percentage of students group by specialization.

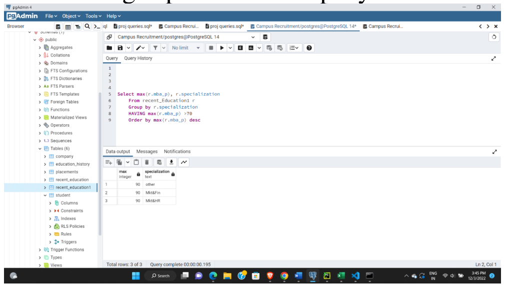


```
-- inner join s.student_id = r.student_id Group by r.specialization Order by r.mba_p desc;
Give the name and id of students with mba_per <80  that got placed in company dell
Select s.student_id, s.student_name From Student s, recent_Education r, Company c, Placements p
Where s.student_id = r.student_id and s.student_id = p.student_id and c.company_id=p.company_id and r.mba_p<80
and c.company_name = 'dell';
```

```
    student_id    student_name
    [PK] integer  text
1             32  Nicholas
2             33  Eric
3             36  Larry
4             82  Gabriel
5             91  Roy
6            120  Amanda
7            141  Christine
8            152  Olivia
9            169  Teresa
10           180  Judy
11           188  Beverly
12           201  vinay
13           214  harini
14           256  Kaaren
15           269  Kael

Total rows: 31 of 31    Query complete 00:00:00.204
```

Fig. 3.

3) Query to find names and id's of student who got less than 80% in MBA and got placed in company Dell.

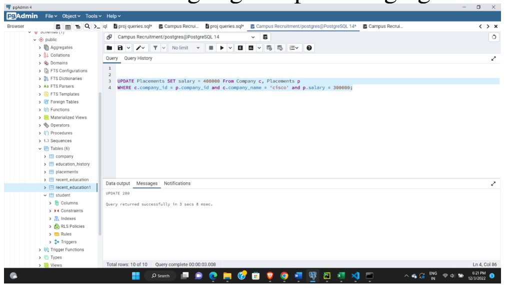


```
Select max(r.mba_p), r.specialization
    From recent_Education1 r
    Group by r.specialization
    HAVING max(r.mba_p) >70
    Order by max(r.mba_p) desc
```

```
   max      specialization
   integer  text
1       90  other
2       90  Mkt&Fin
3       90  Mkt&HR

Total rows: 3 of 3    Query complete 00:00:00.195
```

Fig. 4.

4) Query to give the maximum percentage of students in each specialization having highest percentage greater than 70.



```
UPDATE Placements SET salary = 400000 From Company c, Placements p
WHERE c.company_id = p.company_id and c.company_name = 'cisco' and p.salary = 300000;
```

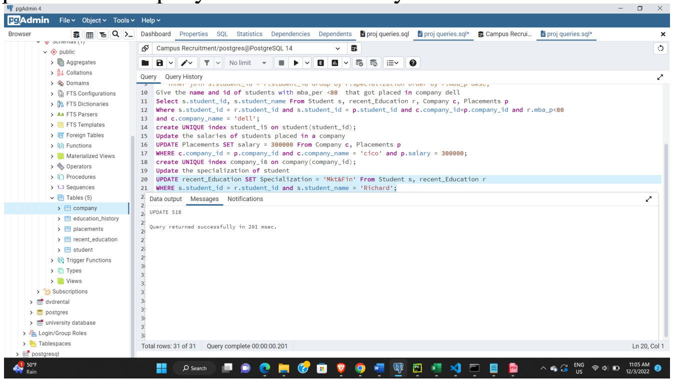
```
UPDATE 280

Query returned successfully in 3 secs 8 msec.
```

Total rows: 10 of 10   Query complete 00:00:03.008   Ln 4, Col 86

Fig. 5.

B. Update Queries

5) Query to update salaries to 400k for the students who got placed in company Cisco with salary 300k.



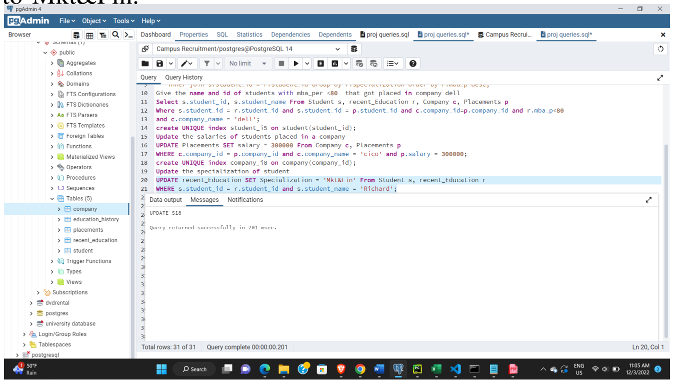
```
Give the name and id of students with mba_per <80  that got placed in company dell
Select s.student_id, s.student_name From Student s, recent_Education r, Company c, Placements p
Where s.student_id = r.student_id and s.student_id = p.student_id and c.company_id=p.company_id and r.mba_p<80
and c.company_name = 'dell';
create UNIQUE index student_i5 on student(student_id);
Update the salaries of students placed in a company
UPDATE Placements SET salary = 300000 From Company c, Placements p
WHERE c.company_id = p.company_id and c.company_name = 'cico' and p.salary = 300000;
create UNIQUE index company_i8 on company(company_id);
Update the specialization of student
UPDATE recent_Education SET Specialization = 'Mkt&Fin' From Student s, recent_Education r
WHERE s.student_id = r.student_id and s.student_name = 'Richard';
```

```
UPDATE 518

Query returned successfully in 201 msec.
```

Total rows: 31 of 31   Query complete 00:00:00.201   Ln 20, Col 1

Fig. 6.

6) Query to update specialization of a student named Richard to Mkt&Fin.



```
Give the name and id of students with mba_per <80  that got placed in company dell
Select s.student_id, s.student_name From Student s, recent_Education r, Company c, Placements p
Where s.student_id = r.student_id and s.student_id = p.student_id and c.company_id=p.company_id and r.mba_p<80
and c.company_name = 'dell';
create UNIQUE index student_i5 on student(student_id);
Update the salaries of students placed in a company
UPDATE Placements SET salary = 300000 From Company c, Placements p
WHERE c.company_id = p.company_id and c.company_name = 'cico' and p.salary = 300000;
create UNIQUE index company_i8 on company(company_id);
Update the specialization of student
UPDATE recent_Education SET Specialization = 'Mkt&Fin' From Student s, recent_Education r
WHERE s.student_id = r.student_id and s.student_name = 'Richard';
```

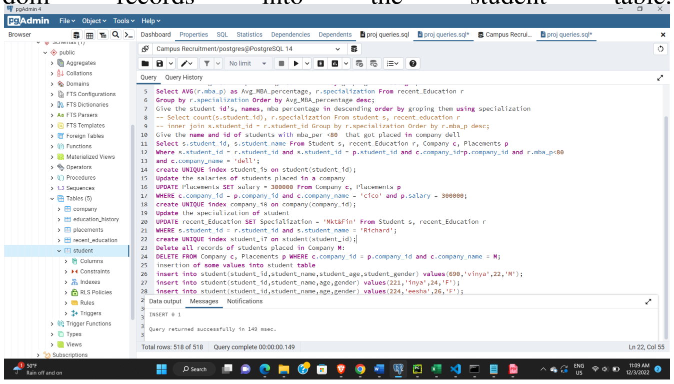
```
UPDATE 518

Query returned successfully in 201 msec.
```

Total rows: 31 of 31   Query complete 00:00:00.201   Ln 20, Col 1

Fig. 7.

C. Insertion Queries

7) Query for insertion of some random records into the student table.



```
Select AVG(r.mba_p) as Avg_MBA_percentage, r.specialization From recent_Education r
Group by r.specialization Order by Avg_MBA_percentage desc;
Give the student id's, names, mba percentage in descending order by groping them using specialization
-- Select count(s.student_id), r.specialization From student s, recent_education r
-- inner join s.student_id = r.student_id Group by r.specialization Order by r.mba_p desc;
Give the name and id of students with mba_per <80  that got placed in company dell
Select s.student_id, s.student_name From Student s, recent_Education r, Company c, Placements p
Where s.student_id = r.student_id and s.student_id = p.student_id and c.company_id=p.company_id and r.mba_p<80
and c.company_name = 'dell';
create UNIQUE index student_i5 on student(student_id);
Update the salaries of students placed in a company
UPDATE Placements SET salary = 300000 From Company c, Placements p
WHERE c.company_id = p.company_id and c.company_name = 'cico' and p.salary = 300000;
create UNIQUE index company_i8 on company(company_id);
Update the specialization of student
UPDATE recent_Education SET Specialization = 'Mkt&Fin' From Student s, recent_Education r
WHERE s.student_id = r.student_id and s.student_name = 'Richard';
create UNIQUE index student_i7 on student(student_id);
Delete all records of students placed in Company M:
DELETE FROM Company c, Placements p WHERE c.company_id = p.company_id and c.company_name = M;
insertion of some values into student table
insert into student(student_id,student_name,student_age,student_gender) values(690,'vinya',22,'M');
insert into student(student_id,student_name,age,gender) values(221,'inya',24,'F');
insert into student(student_id,student_name,age,gender) values(224,'eesha',26,'F');
```

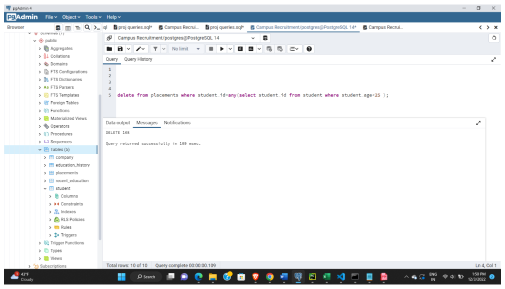
```
INSERT 0 1

Query returned successfully in 149 msec.
```

Total rows: 518 of 518   Query complete 00:00:00.149   Ln 22, Col 55

Fig. 8.

D. Deletion Queries

8) Query to delete records of student from placements who age is less than 25.



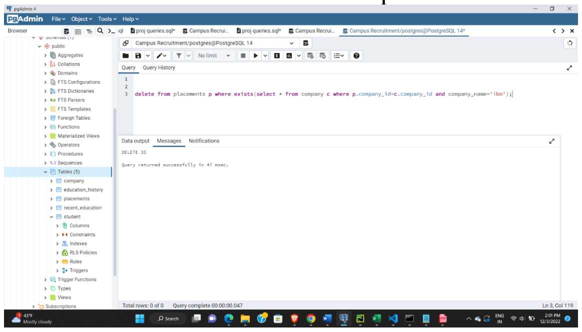
```
delete from placements where student_id=any(select student_id from student where student_age<25 );
```

```
DELETE 168

Query returned successfully in 189 msec.
```

Total rows: 10 of 10   Query complete 00:00:00.189   Ln 4, Col 1

Fig. 9.

9) Query to delete record of students who have been place in IBM.



```
delete from placements p where exists(select * from company c where p.company_id=c.company_id and company_name='ibm');
```

```
DELETE 35

Query returned successfully in 47 msec.
```

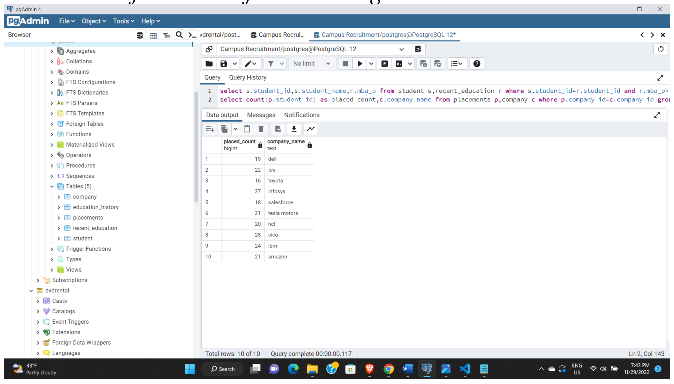
Total rows: 0 of 0   Query complete 00:00:00.047   Ln 3, Col 119

Fig. 10.

VI. Indexing

Upon the execution of various queries on the relations in the implemented database, to further improve the performance we have used indexing. After indexing we have noticed a reduction in the query execution time. The following are some examples implying the reduction of query execution time upon indexing.

Example 1: The execution time of the below query has reduced from 117ms to 62ms. Below are the screenshots of the same before and after indexing.

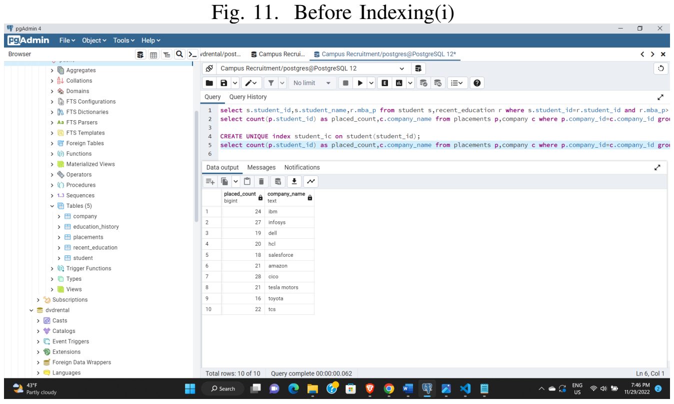


```
select s.student_id,s.student_name,r.mba_p from student s,recent_education r where s.student_id=r.student_id and r.mba_p>
select count(p.student_id) as placed_count,c.company_name from placements p,company c where p.company_id=c.company_id gro
```

	placed_count bigint	company_name text
1	19	dell
2	22	tcs
3	16	toyota
4	27	infosys
5	18	salesforce
6	21	tesla motors
7	20	hcl
8	28	cico
9	24	ibm
10	21	amazon

Total rows: 10 of 10 Query complete 00:00:00.117 Ln 2, Col 143

Fig. 11. Before Indexing(i)



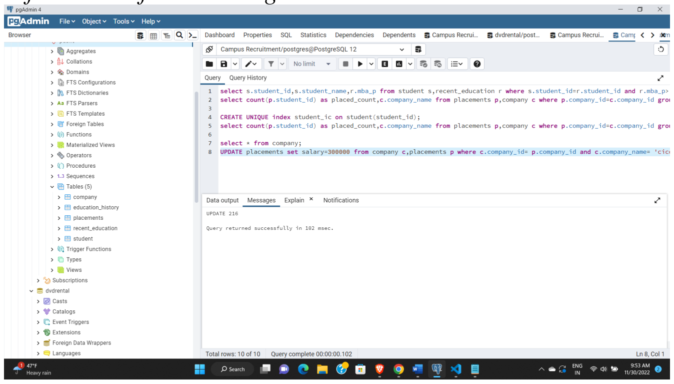
```
select s.student_id,s.student_name,r.mba_p from student s,recent_education r where s.student_id=r.student_id and r.mba_p>
select count(p.student_id) as placed_count,c.company_name from placements p,company c where p.company_id=c.company_id gro

CREATE UNIQUE index student_ic on student(student_id);
select count(p.student_id) as placed_count,c.company_name from placements p,company c where p.company_id=c.company_id gro

```

	placed_count bigint	company_name text
1	24	ibm
2	27	infosys
3	19	dell
4	20	hcl
5	18	salesforce
6	21	amazon
7	28	cico
8	21	tesla motors
9	16	toyota
10	22	tcs

Total rows: 10 of 10 Query complete 00:00:00.062 Ln 6, Col 1

Fig. 12. After Indexing(i)

Example 2: The execution time of the below query has reduced from 102ms to 42ms. Below are the screenshots of the same before and after indexing.

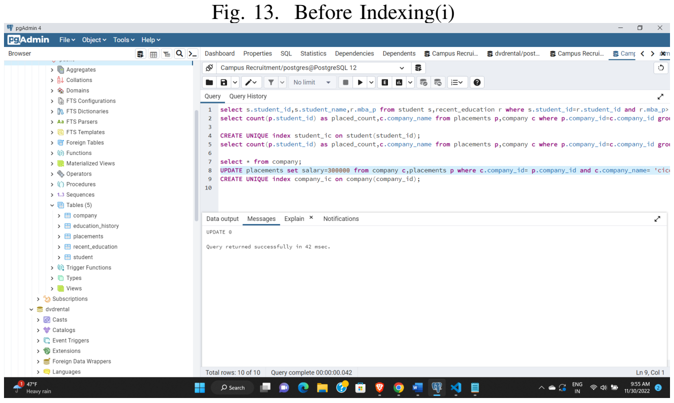


```
select s.student_id,s.student_name,r.mba_p from student s,recent_education r where s.student_id=r.student_id and r.mba_p>
select count(p.student_id) as placed_count,c.company_name from placements p,company c where p.company_id=c.company_id gro

CREATE UNIQUE index student_ic on student(student_id);
select count(p.student_id) as placed_count,c.company_name from placements p,company c where p.company_id=c.company_id gro

select * from company;
UPDATE placements set salary=300000 from company c,placements p where c.company_id= p.company_id and c.company_name= 'cic
```

```
UPDATE 216

Query returned successfully in 102 msec.
```

Total rows: 10 of 10 Query complete 00:00:00.102 Ln 8, Col 1

Fig. 13. Before Indexing(i)



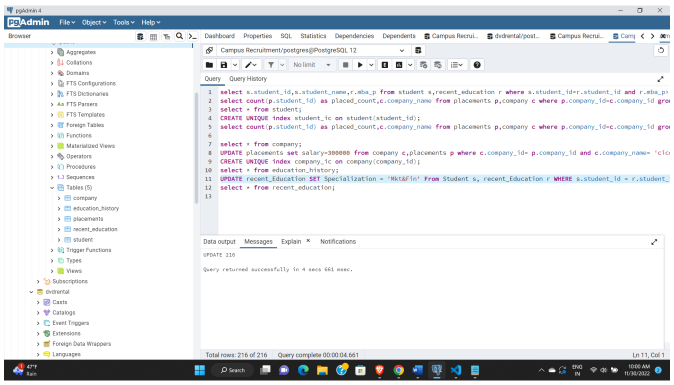
```
select s.student_id,s.student_name,r.mba_p from student s,recent_education r where s.student_id=r.student_id and r.mba_p>
select count(p.student_id) as placed_count,c.company_name from placements p,company c where p.company_id=c.company_id gro

CREATE UNIQUE index student_ic on student(student_id);
select count(p.student_id) as placed_count,c.company_name from placements p,company c where p.company_id=c.company_id gro

select * from company;
UPDATE placements set salary=300000 from company c,placements p where c.company_id= p.company_id and c.company_name= 'cic
CREATE UNIQUE index company_ic on company(company_id);

```

```
UPDATE 0

Query returned successfully in 42 msec.
```

Total rows: 10 of 10 Query complete 00:00:00.042 Ln 8, Col 1

Fig. 14. After Indexing(i)

Example 3: The execution time of the below query has reduced from 661ms to 48ms. Below are the screenshots of the same before and after indexing.

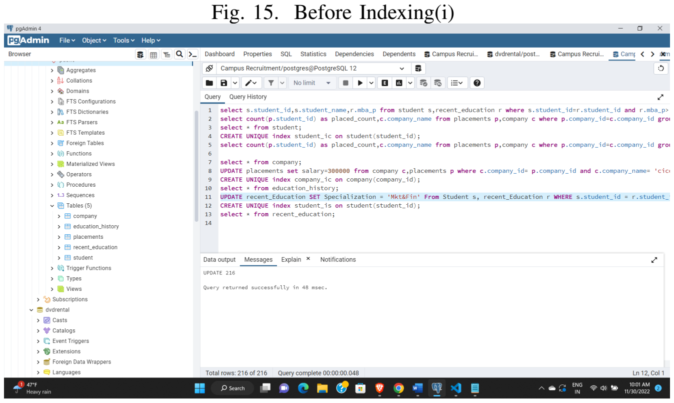


```
select s.student_id,s.student_name,r.mba_p from student s,recent_education r where s.student_id=r.student_id and r.mba_p>
select count(p.student_id) as placed_count,c.company_name from placements p,company c where p.company_id=c.company_id grou
select * from student;
CREATE UNIQUE index student_ic on student(student_id);
select count(p.student_id) as placed_count,c.company_name from placements p,company c where p.company_id=c.company_id grou

select * from company;
UPDATE placements set salary=300000 from company c,placements p where c.company_id= p.company_id and c.company_name= 'cic
CREATE UNIQUE index company_ic on company(company_id);
select * from education_history;
UPDATE recent_Education SET Specialization = 'Mkt&Fin' From Student s, recent_Education r WHERE s.student_id = r.student_
select * from recent_education;
```

```
UPDATE 216

Query returned successfully in 4 secs 661 msec.
```

Total rows: 216 of 216 Query complete 00:00:04.661

Fig. 15. Before Indexing(i)



```
select s.student_id,s.student_name,r.mba_p from student s,recent_education r where s.student_id=r.student_id and r.mba_p>
select count(p.student_id) as placed_count,c.company_name from placements p,company c where p.company_id=c.company_id grou
select * from student;
CREATE UNIQUE index student_ic on student(student_id);
select count(p.student_id) as placed_count,c.company_name from placements p,company c where p.company_id=c.company_id grou

select * from company;
UPDATE placements set salary=300000 from company c,placements p where c.company_id= p.company_id and c.company_name= 'cic
CREATE UNIQUE index company_ic on company(company_id);
select * from education_history;
UPDATE recent_Education SET Specialization = 'Mkt&Fin' From Student s, recent_Education r WHERE s.student_id = r.student_
CREATE UNIQUE index student_is on student(student_id);
select * from recent_education;
```

```
UPDATE 216

Query returned successfully in 48 msec.
```

Total rows: 216 of 216 Query complete 00:00:00.048

Fig. 16. After Indexing(i)

VII. Team Contributions

1) Sankeerthana:
   - Problem statement selection
   - Design overview of the database
   - ER diagram implementation
2) Shreya:
   - Database Creation
   - Cleaning and modification on dataset
   - Record Insertion into relations
3) Abhigna:
   - Writing queries
   - Testing queries
   - Indexing
4) Report Generation : Sankeerthana,Shreya,Abhigna